AMENDMENTS TO THE SPECIFICATION

Please amend the specification as follows. All references are to the Application as filed.

Replace the paragraph starting on line 18, page 4, with the following paragraph.

According the present invention, organization and decompression of data ~~segment~~ segments received by the printer is done by as many as three tasks in the printer software - - a transform task, a command handler, and a decompression task. If the decompression is performed by printer hardware, the decompression software task is not needed and only the transform task and command handler are employed. The transform task receives the print~~er~~ header and passes the appropriate information to the command handler. The command handler then waits for all of the data to be received and decompressed. As each segment is received by the link, it is stored in a memory buffer. The transform task allocates a memory partition for the decompressed data and sends a message to either the decompression task or printer hardware, asking for the data to be decompressed. When the decompression task is finished with each segment, it sends a message to the command handler informing it that the data is available. Then, based on information from the print header, the command handler assembles the data and starts the printing process.

Replace the paragraph starting on line 14, page 5, with the following paragraph.

Fig. 3 is an illustration of [[a]] full data streams containing equal amounts of data according to the present invention.

Replace the paragraph starting on line 10, page 6, with the following paragraph.

Referring now to FIG. 1, a block diagram of a system for transmitting and processing multiple print data streams according to the present invention will be described. The system 100 comprises a host 110 on which host software 150 is resident. The host 110 can be a stand-alone personal computer (PC), a print server, or other types of networked computers. Host software 150 performs pre-processing of the print data and directs transmission of the data through the data link 120 to the print device 130. The data link 120 may be a universal serial bus (USB) or

any other ~~any~~ known means of linking hardware devices for data transmission. The print device 130 may be an inkjet printer or other type of printing device. The print device 130 contains firmware 140 and hardware 160, which further processes print data received from the host 110 to enable the print device 130 to produce a printed item, such as a multi-color graph or plain text document. The manner in which the components 110, 120, 130, 140, 150 and 160 of system 100 may be assembled is well known and will not be described further.

Replace the paragraph starting on line 4, page 7, with the following paragraph.

The print data in each stream is then further divided into smaller segments at step 220. The host software 150 then uses a compression algorithm to compress each data segment at step 230. The compression of the data segments at step 230 can be performed using a single compression algorithm for each segment, or alternatively host software 150 can employ multiple compression algorithms, the compression algorithm for each segment being selected by the host software 150 so as to maximize system efficiency and performance. Those in the art will understand that the method and system of the present invention may be employed without compressing the individual data segments. In such a case, the functionality of the system that is directed towards handling compressed data segments is simply not used and the uncompressed data segments are treated as decompressed data segments would be by the other components of the system.

Replace the paragraph starting on line 17, page 7, with the following paragraph.

Host software then creates a header for each compressed data segment at step 240. The header describes the size of the data in compressed and uncompressed form, and that information ~~that~~ is later used by the printer firmware 140 to further process the data.

Replace the paragraph starting on line 8, page 8, with the following paragraph.

After the segmentation, compression and ordering information is determined by the host software ~~host~~ 150, the print header is then sent to the print device 130 at step 260, followed by the data segments at step 270. The data segments may be sent to the print device in many different sequences for any particular print job. Ordering of segments for transmission is independent of the color composition of each swath and the direction (left to right or right to

left). The following examples demonstrate how the segment ordering could be handled for a variety of printing situations.

Replace the paragraph starting on line 1, page 9, with the following paragraph.

However, practical application of the present invention frequently requires the printing of swaths, which are not composed of equal amounts of each color. Rather, the creation of a swath requires the printing device 130 to apply each color at different points on a page. Referring now to FIG. 4, an example is shown of how data segments may be ordered when each data stream contains unequal amounts of print data. That is, the colors to be applied to produce the print swath start and end at [[a]] different points in the swath. Assuming left to right printing, the print data ~~need~~ needed to produce the swath is broken down into a cyan data stream 410, a magenta stream 420 and a yellow stream 430. The swath to be printed is at first exclusively magenta, then magenta in combination with cyan and yellow. Accordingly, cyan data stream 410 is empty at section 415 and the yellow data stream 430 is empty at section 435. Only magenta data stream 420 contains data at the front section of the stream, as is represented by segment 421. One possible resulting order of the data streams for transmission is M1, Y1, C1, M2, Y2, C2, M3, and Y3. The data-less portions 415 and 435 of data streams 410 and 430 are empty – no padding of the data is required. The printer firmware 140 uses the firegroup count and offset data 450 contained in the print header to calculate the beginning and ending points for application of each color on the page. These numbers describe the starting position of each color as a distance from the starting position (offset) and the amount of data that will be sent for each color (firegroup count). By using these numbers along with the starting position of the swath, the printer firmware 140 can determine when to start firing the printheads and when to send the print data to each colorhead.

Replace the paragraph starting on line 22, page 9, with the following paragraph.

Referring now to FIG. 5, an example of data segment ordering is shown for the situation where multiple data streams are mostly empty. Again, assuming left to right printing, cyan data stream 510 is mostly full and contains three segments 511, 512 and 513. The particular swath to be printed does not require magenta or yellow until near the end of the swath, and thus magenta data stream 520 and yellow data stream 530 are mostly empty at their front sections and contain

only single data segments 521 and 522 ~~531~~, respectively. In this situation, the segments would be sent in the order C1, C2, C3, M1, and Y1. The empty sections 515, 525 and 535 of each data stream are not padded with zeroes or any other form of placeholder data. Rather, the printer firmware 140 is capable of re-assembling the data segments into streams using the starting position and offset information contained in the print header. FIG. 5 illustrates two significant advantages of the present invention. In this example, where significant portions of two data streams are empty, the lack of a need for data padding substantially reduces the amount of traffic over data link 120. The lack of padding also reduces the amount of memory required in printing device 130 to store print data during processing. Because no memory is expended on storing padding, more print data may be buffered at any one time, creating added economy of memory resources.

Replace the paragraph starting on line 13, page 10, with the following paragraph.

Print data is transferred from memory to the printhead(s) of the printing device 130 through direct memory access (DMA). As the printhead passes across the page, it is important that the stream of data to the printhead not be interrupted. According to the present invention, the continuous data stream is accomplished by providing the DMA controller with registers that allow sequential DMA transfers to be set up all at once. When first setting up the DMA to a printhead, the first data segment is setup along with pointers to the second data segment. The system of the present invention automatically switches from the first to the second upon the end of the first data segment. The memory space previously occupied by the first segment is immediately de-allocated for re-use and filled by another segment of print data. The printer firmware 140 then has until the second segment completes to setup pointers for the third ~~segment~~ segment, and so on. This process continues for every data segment in the swath, so that data segments are continuously pulled from and allocated to printer memory until the print job is complete.

Replace the paragraph starting on line 21, page 11, with the following paragraph. All changes are shown relative to the Application as filed.

Additional benefits can be obtained by aligning the segments with the DMA FIFOs in the printer ASIC so that DMA transfers for all data streams can be completed at one time, lessening the time requirements by reducing the interrupt handler overhead. Referring now to Figures 8A and 8B ~~Figure 8~~, the advantages of coordinating DMA interrupts are shown by illustrating the consequence of two methods of scheduling DMA interrupts, one without segment alignment and one with segment alignment. In Method 1, illustrated in FIG. 8A, DMA interrupts occur at each segment boundary in each color. So, there are nine DMA interrupts: one at the start of cyan 811, one at the start of 831 yellow, one at segment 812 cyan, two between segments 821 and 822 of magenta, two between segments of yellow (832 and 833), one at the end of 813 cyan, and one at the end of 823 magenta. In Method 2, illustrated in FIG. 8B, DMA interrupts occur at the start and end of colors and at coordinated points in the data streams. Therefore, there are six interrupts in Method 2: one at the start of 841 cyan, one at the start of 861 yellow, two between color segments 862 and 863, one at the end of 843 cyan, and one at the end of 853 magenta. The two interrupts between color segments handle the DMA interrupts for all three colors at the same time. This gives a maximum of six DMA interrupts per swath. By eliminating the number of DMA interrupts, the printer firmware spends less time managing the print DMA system and can use that time for other processing, such as data decompression. Reducing the number of interrupts also eases the burden on the interrupt controller and allows other interrupts in the system more opportunity to be serviced.

Replace the paragraph starting on line 22, page 12, with the following paragraph.

Referring now to FIG. 9, the functions performed by a transform task in handling multiple compressed data streams are illustrated in a flow chart. The transform task 900 waits for data to be received at step ~~920~~ 910. If print data is received at step ~~930~~ 920, the transform task 900 determines whether the data is a print header at step 940, a data segment at step 950, or an end print command at step 960. If the data is a print header, the header is stored at step 941 and a message is sent to the command handler task about the upcoming swath at step 942. If the data is a data segment, the transform task 900 allocates memory space for data storage and DMA data at step 951, then sends a message to the decompression task to decompress the segment at

step 952. The transform task allocates a memory partition for each segment that correctly matches the size of the segment using the information contained in the segment header, thus preventing fragmentation of the print device memory. If the data is an end print command, the transform task 900 sends a message to the command handler to complete handling of the swath print data at step 961.

Replace the paragraph starting on line 13, page 13, with the following paragraph.

Referring now to FIG. 10, the functions performed by the decompression task are illustrated with a flow chart. The decompression task 1000 waits for a message step 1010. When a data segment is received, decompression task 1000 determines if the data is compressed at step 1020. If the data is compressed, a determination is made as to which compression algorithm is employed at step 1021, and the segment is decompressed using that algorithm at step 1022. Decompression task 1000 also determines if the data has a checksum at step 1030, and if so, performs a checksum validation at step 1031. After decompression and checksum validation, the decompression task 1000 sends a message with a pointer and the size of the processed data segment to the command handler 1040. It will be understood by those in the ~~ART~~ art that the decompression of data segments may also be performed by printer hardware. In such cases, the printer hardware 160 would perform the above-described functions of the decompression task 1000.